CS4442/9542b: Artificial Intelligence II
Prof. Olga Veksler

Lecture 2: Machine Learning

Introduction to ML

Basic Linear Algebra

Matlab

Some slides on Linear Algebra are from Patrick Nichols

Outline

- Introduction to Machine Learning
- Basic Linear Algebra
- Matlab Intro

Intro: What is Machine Learning?

- How to write a computer program that automatically improves its performance through experience
- Machine learning is useful when it is too difficult to come up with a program to perform a desired task
- Make computer to learn by showing examples (most frequently with correct answers)
  - "supervised" learning or learning with a teacher
- In practice: computer program (or function) which has a tunable parameters, tune parameters until the desirable behavior on the examples

Different Types of Learning

- **Learning from examples:**
  - **Supervised Learning**: given training examples of inputs and corresponding outputs, produce the "correct" outputs for new inputs
  - **Unsupervised Learning**: given only inputs as training, find structure in the world: e.g. discover clusters
- **Reinforcement Learning** (similar to animal learning): an agent takes inputs from the environment, and takes actions that affect the environment. Occasionally, the agent gets a reward or punishment. The goal is to learn to produce action sequences that maximize the expected reward (e.g. driving a robot without bumping into obstacles). Not covered in this course.

slide is modified from Y. LeCun

Learning is NOT Memorization

- rote learning is easy
- Say we have 2 classes: face and non-face images
- memorize all the "face" examples
- For a new image, see if it is present in the stored "face" collection
  - if yes, output "face" as the classification result
  - If no, output "non-face"
- PROBLEM: in general, new "face" images are different from stored "face" examples
- The ability to produce correct outputs or behavior on previously unseen inputs is called GENERALIZAITION
- Rote learning is memorization without generalization
- The big question of Learning Theory (and practice): how to get good generalization with a limited number of examples

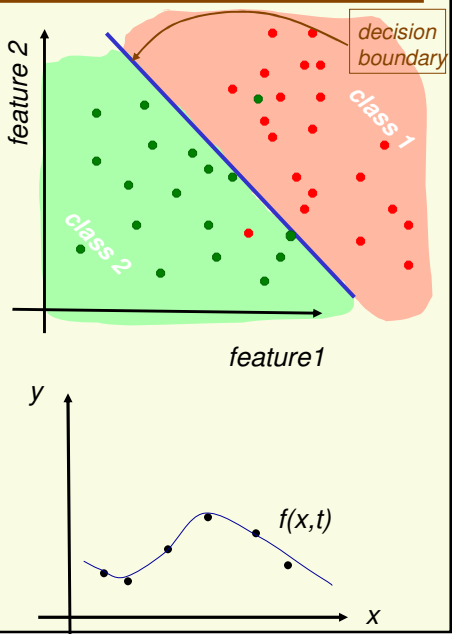
slide is modified from Y. LeCun

Sketch of Machine Learning *(supervised)*

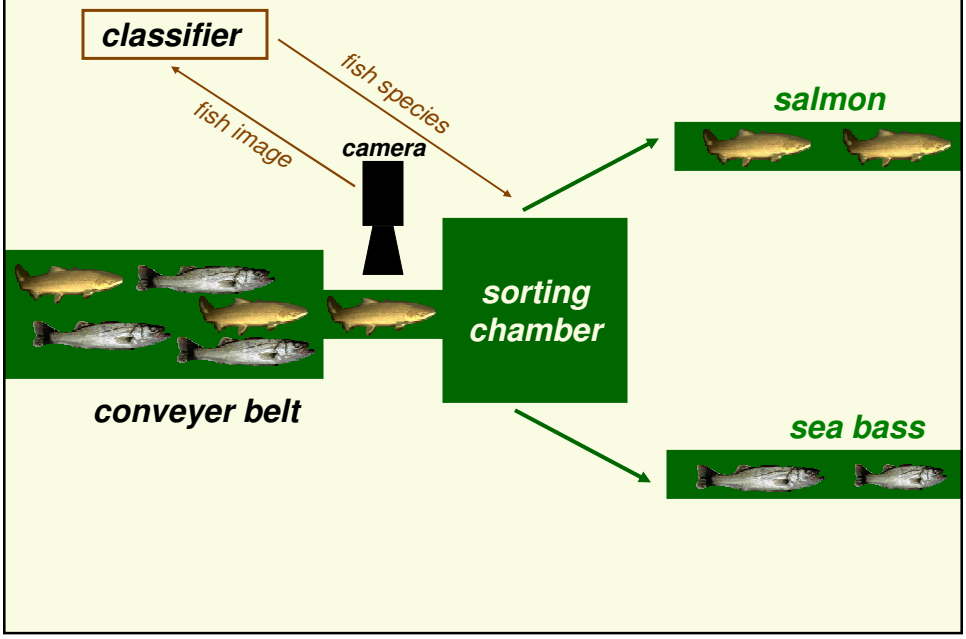
- Training samples (or examples) $X^1, X^2, \ldots X^n$
- Each example is typically multi-dimensional
  - $X^i_1, X^i_2, \ldots, X^i_d$ are typically called *features*, X^i is sometimes called a *feature vector*
  - **How many features and which features do we take?**
- Know desired output for each example (labeled samples) $Y^1, Y^2, \ldots Y^n$
  - This learning is supervised ("teacher" gives desired outputs).
  - Y^i are often one-dimensional, but can be multidimensional
- Two types of supervised learning:
  - Classification (we will only do classification in this course):
    - Y^i takes value in finite set and typically called a *label* or a *class*
    - Example: $Y \in \{\text{sunny,cloudy,raining}\}$
  - Regression, or function fitting:
    - Y^i continuous. In this case, it is typically called an *output value*
    - Example: Y=temperature $\in [-60,60]$

Two types of Machine Learning

1. **Classification** (mostly deal with in this course)
   - outputs y_i are discrete, represent categories (ex.: object categories face, car, etc.)
   - Usually visualize decision regions and decision boundary
   - f(x,t) is usually called classifier
2. **Regression:**
   - outputs y_i are continuous, example: temperature
   - This is also called "curve fitting"



Our Toy Application: fish sorting



How to design a Classification system?

- Collect data and classify by hand

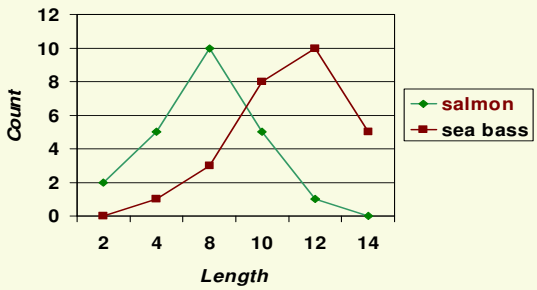
- Preprocess by segmenting fish from background

- Extract possibly discriminating features
  - length, lightness,width,number of fins,etc.
- Classifier design
  - Choose model for classifier
  - Train classifier on part of collected data (training data)
- Test classifier on the rest of collected data (test data) i.e. the data not used for training
  - Should classify new data (new fish images) well

Classifier design

- Notice salmon tends to be shorter than sea bass
- Use *fish length* as the discriminating feature
- Count number of bass and salmon of each length

	2	4	8	10	12	14
bass	0	1	3	8	10	5
salmon	2	5	10	5	1	0



Fish length as discriminating feature

- Find the best length ***L*** threshold

fish length < L ⇩ ***classify as salmon***

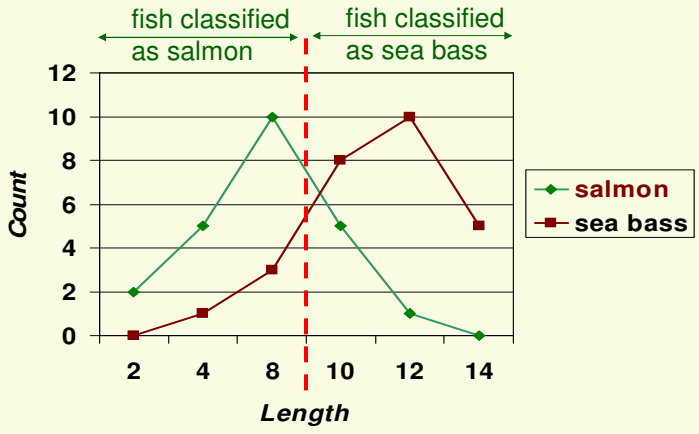
fish length > L ⇩ ***classify as sea bass***

- For example, at ***L*** = 5, misclassified:
  - 1 sea bass
  - 16 salmon

	2	4	8	10	12	14
bass	0	1	3	8	10	5
salmon	2	5	10	5	1	0

- Classification error (total error): $\frac{17}{50} = 34\%$

Fish Length as discriminating feature



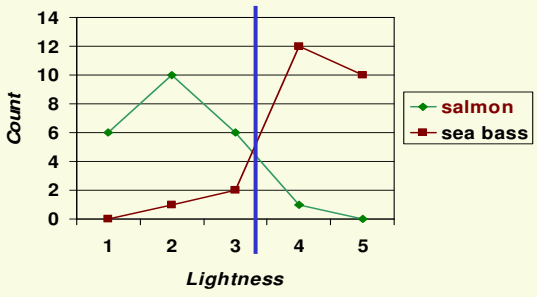
- After searching through all possible thresholds ***L***, the best ***L***= 9, and still 20% of fish is misclassified

Next Step

- Lesson learned:
  - Length is a poor feature alone!
- What to do?
  - Try another feature
  - Salmon tends to be lighter
  - Try average fish lightness

Fish lightness as discriminating feature

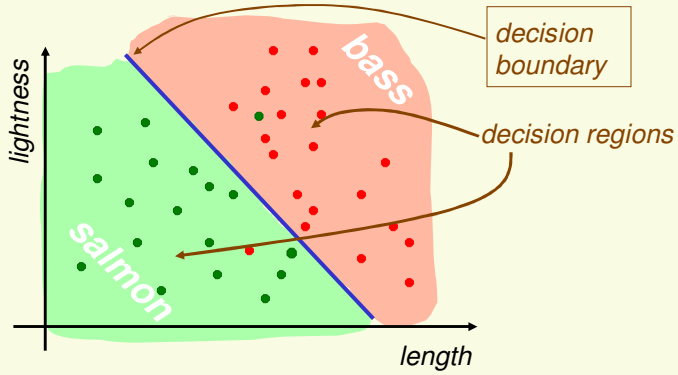
	1	2	3	4	5
bass	0	1	2	10	12
salmon	6	10	6	1	0



- Now fish are well separated at lightness threshold of 3.5 with classification error of 8%

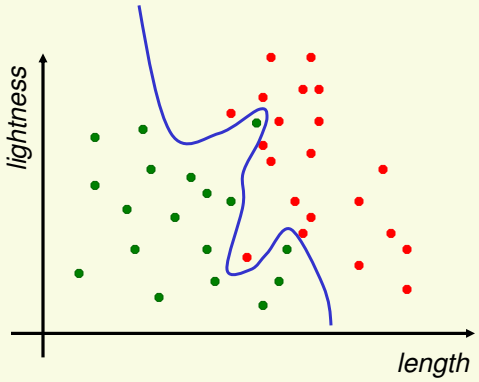
Can do even better by feature combining

- Use both *length* and *lightness* features
- Feature vector [*length*,*lightness*]



- Classification error 4%

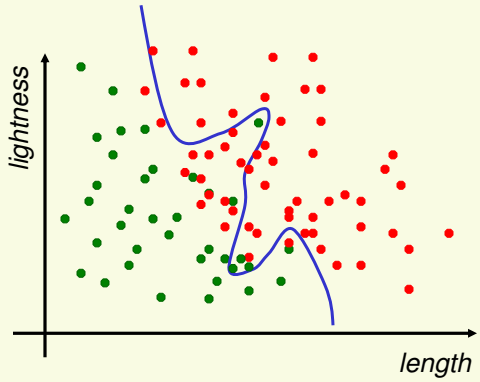
Better decision boundary



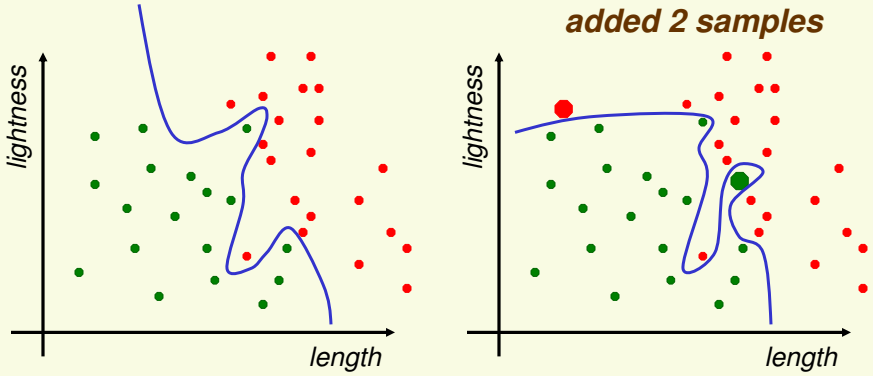
- Ideal decision boundary, 0% classification error

Test Classifier on New Data

- Classifier should perform well on new data
- Test "ideal" classifier on new data: 25% error



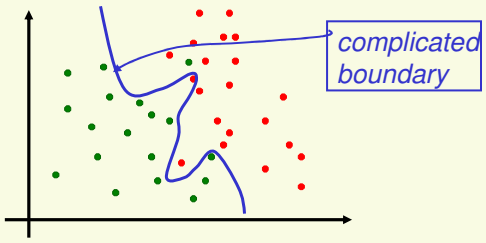
What Went Wrong?



- We always have only a limited amount of data, not all possible data
- We should make sure the decision boundary does not adapt too closely to the data we have at hand, but rather grasps the "big picture"

What Went Wrong?

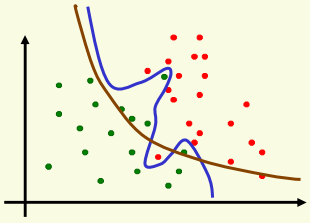
- Poor ***generalization***



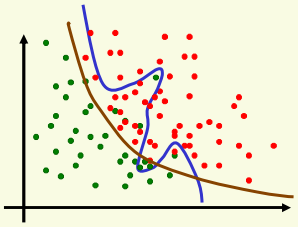
- Complicated boundaries do not generalize well to the new data, they are too "tuned" to the particular training data, rather than some true model which will separate salmon from sea bass well.
  - This is called overfitting the data

Generalization

training data



testing data



- Simpler decision boundary does not perform ideally on the training data but generalizes better on new data
- Favor simpler classifiers
  - William of Occam (1284-1347): "entities are not to be multiplied without necessity"

System Structure

domain dependent

camera, microphones, medical imaging devices, etc.

Patterns should be well separated and should not overlap.

Extract discriminating features. Good features make the work of classifier easy.

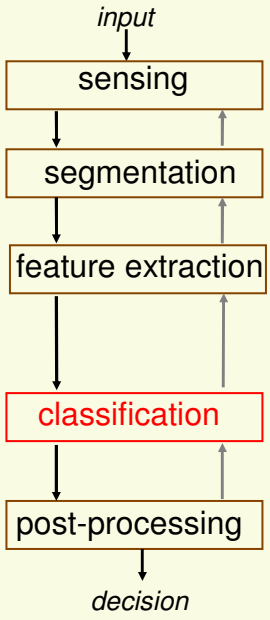
good features: bad features:

Use features to assign the object to a category. Better classifier makes feature extraction easier.

Our main topic in this course

Exploit context (input depending information) to improve system performance

Tne cat → ***The cat***



Sketch of Machine Learning *(supervised)*

- Chose a *learning machine* f(X,W)
  - W are tunable weights
  - X is the input sample
  - f(X,W) should output the correct class of sample X
  - use labeled samples to tune weights W so that f(X,W) give the correct label for sample X

Training and Testing

- There are 2 phases, training and testing
  - Divide all labeled samples $X^1, X^2, \ldots X^n$ into 2 sets, *training* set and *testing* set
  - Training phase is for "teaching" our machine (finding optimal weights W)
  - Testing phase is for evaluating how well our machine works on unseen examples
- Training phase
  - Find the weights W s.t. $f(X^i, W) = Y^i$ "as much as possible" for the *training* samples X^i
  - "as much as possible" needs to be defined
  - Training can be quite complex and time-consuming

Testing

- Testing phase
  - The goal is to design machine which performs well on unseen examples (which are typically different from labeled examples)
  - Evaluate the performance of the trained machine f(X,W) on the testing samples (unseen labeled samples)
  - Testing the machine on unseen labeled examples lets us approximate how well it will perform in practice
  - If testing results are poor, may have to go back to the training phase and redesign f(X,W)

Generalization and Overfitting

- *Generalization* is the ability to produce correct output on previously unseen examples
  - In other words, low error on unseen examples
  - **Good generalization is the main goal of ML**
- Low train error does not necessarily imply that we will have low test error
  - Very easy to produce f(X,W) which is perfect on training samples
    - "memorize" all the training samples and output their correct label
    - random label on unseen examples
    - No training error but horrible test error
- *Overfitting*
  - when the machine performs well on training data but poorly on testing data

Sketch of Machine Learning (supervised)

- None of the stages are easy
- Modeling stage:
  - Which features do we extract from training data (which are usually images in vision). How many features?
- Training stage:
  - Which function f(x,t) do we choose? Has to be expressive enough to model our problem, yet not to complicated to avoid ***overfitting***
  - How do we tweak parameters t to ensure f(x,t) = y for most training samples (x,y) ? This step is usually done by optimization, can be quite expensive.
- Evaluation stage
  - Good performance on the training data does not guarantee good performance on data we haven't seen yet. In fact, no error on training data frequently means that we overfitted to the training data

Basic Linear Algebra Introduction

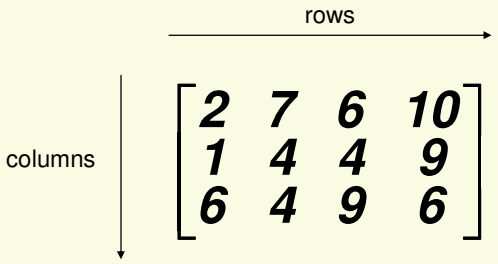
- Basic Concepts in Linear Algebra
  - vectors and matrices
  - products and norms
  - vector spaces and linear transformations
- Introduction to Matlab

Why Linear Algebra?

- For each example (e.g. a fish image), we will extract a set of features (e.g. length, width, color)
- This set of features we will represent as a *feature vector*
  - [length, width, color,…]
- All collected examples will be represented as collection of (feature) vectors
  - $[l_1, w_1, c_1, \ldots]$, $[l_2, w_2, c_2, \ldots]$, $[l_3, w_3, c_3, \ldots]$, …
    example 1 example 2 example 3
- Besides representation, we will often use linear models since they are simple and computationally feasible

What is a Matrix?

- A matrix is a set of elements, organized into rows and columns



Basic Matrix Operations

- Addition, Subtraction, Multiplication by a scalar

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} a+e & b+f \\ c+g & d+h \end{bmatrix}$$ **Just add elements**

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} - \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} a-e & b-f \\ c-g & d-h \end{bmatrix}$$ **Just subtract elements**

$$\alpha \cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} \alpha \cdot a & \alpha \cdot b \\ \alpha \cdot c & \alpha \cdot d \end{bmatrix}$$ **Just multiply every entry**

Matrix Transpose

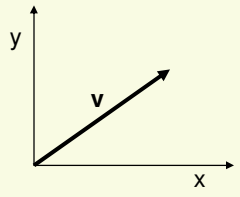
- n by m matrix A and its m by n transpose A^T

$$A = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & \cdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix} \qquad A^T = \begin{bmatrix} x_{11} & x_{21} & \cdots & x_{n1} \\ x_{12} & x_{22} & \cdots & x_{n2} \\ \vdots & \vdots & \cdots & \vdots \\ x_{1m} & x_{2m} & \cdots & x_{nm} \end{bmatrix}$$

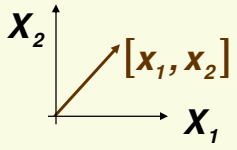
Vectors

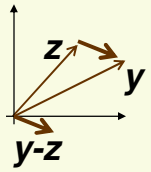
- Vector: N x 1 matrix
- Interpretation: a line in N dimensional space
- Dot Product and Magnitude defined on vectors only

$$\vec{v} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$



More on Vectors



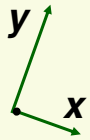
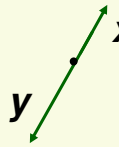


- n-dimensional row vector $\boldsymbol{x} = [x_1 \; x_2 \; \ldots \; x_n]$
- Transpose of row vector is column vector $\boldsymbol{x}^T = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$
- *Vector* product (or *inner* or *dot* product)

$$\langle x, y \rangle = x \cdot y = x^T y = x_1 y_1 + x_2 y_2 + \ldots + x_n y_n = \sum_{i=1..n} x_i y_i$$

More on Vectors

- *Euclidian norm* or *length* $\|\boldsymbol{x}\| = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle} = \sqrt{\sum_{i=1..n} x_i^2}$
- If $\|\boldsymbol{x}\|=1$ we say $\boldsymbol{x}$ is *normalized* or *unit* length
- angle θ between vectors $\boldsymbol{x}$ and $\boldsymbol{y}$: $\cos\theta = \dfrac{\boldsymbol{x}^T \boldsymbol{y}}{\|\boldsymbol{x}\| \|\boldsymbol{y}\|}$

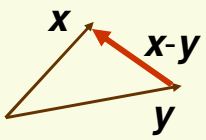
 $\cos\theta = 0$ $\boldsymbol{x}^T \boldsymbol{y} = 0$ x orthogonal to y $\boldsymbol{x} \perp \boldsymbol{y}$	 $\cos\theta = 1$ $\boldsymbol{x}^T \boldsymbol{y} = \|\boldsymbol{x}\| \|\boldsymbol{y}\| > 0$	 $\cos\theta = -1$ $\boldsymbol{x}^T \boldsymbol{y} = -\|\boldsymbol{x}\| \|\boldsymbol{y}\| < 0$

- Thus inner product captures direction relationship between $\boldsymbol{x}$ and $\boldsymbol{y}$

More on Vectors

- Vectors x and y are orthonormal if they are orthogonal and ||x||=||y||=1
- Euclidian distance between vectors x and y

$$\|\boldsymbol{x}-\boldsymbol{y}\| = \sqrt{\sum_{i=1..n}(\boldsymbol{x}_i-\boldsymbol{y}_i)^2}$$



Linear Dependence and Independence

- Vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ are linearly dependent if there exist constants $\alpha_1, \alpha_2, \ldots, \alpha_n$ s.t.
  1. $\alpha_1 \boldsymbol{x}_1 + \alpha_2 \boldsymbol{x}_2 + \ldots + \alpha_n \boldsymbol{x}_n = \boldsymbol{0}$
  2. at least one $\alpha_i \neq \boldsymbol{0}$
- Vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ are linearly independent if $\alpha_1 \boldsymbol{x}_1 + \alpha_2 \boldsymbol{x}_2 + \ldots + \alpha_n \boldsymbol{x}_n = \boldsymbol{0} \Rightarrow \alpha_1 = \ldots = \alpha_n = \boldsymbol{0}$

Vector Spaces and Basis

- The set of all n-dimensional vectors is called a vector space $\boldsymbol{V}$
- A set of vectors $\{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n\}$ are called a basis for vector space if any $\boldsymbol{v}$ in $\boldsymbol{V}$ can be written as $\boldsymbol{v} = \alpha_1 \boldsymbol{u}_1 + \alpha_2 \boldsymbol{u}_2 + \ldots + \alpha_n \boldsymbol{u}_n$
- $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n$ are independent implies they form a basis, and vice versa
- $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n$ give an orthonormal basis if
  1. $|\boldsymbol{u}_i| = 1 \quad \forall \boldsymbol{i}$
  2. $\boldsymbol{u}_i \perp \boldsymbol{u}_j \quad \forall \boldsymbol{i} \neq \boldsymbol{j}$

Orthonormal Basis

$$x = [1 \quad 0 \quad 0]^T \qquad x \cdot y = 0$$
$$y = [0 \quad 1 \quad 0]^T \qquad x \cdot z = 0$$
$$z = [0 \quad 0 \quad 1]^T \qquad y \cdot z = 0$$

X, Y, Z is an orthonormal basis. We can describe any 3D point as a linear combination of these vectors.

Matrix Product

$$AB = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1d} \\ \vdots & \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nd} \end{bmatrix} \begin{bmatrix} b_{11} & \cdots & b_{1m} \\ b_{21} & \cdots & b_{2m} \\ b_{31} & \cdots & b_{3m} \\ \vdots & \cdots & \vdots \\ b_{d1} & \cdots & b_{dm} \end{bmatrix} = \begin{bmatrix} & c_{ij} & \end{bmatrix} = C$$

$$c_{ij} = \langle a^i, b_j \rangle$$

a^i is row i of A
b_j is column j of B

- # of columns of A = # of rows of B
- even if defined, in general $AB \neq BA$

Matrices

- **Rank** of a matrix is the number of linearly independent rows (or equivalently columns)
- A square matrix is **non-singular** if its rank equal to the number of rows. If its rank is less than number of rows it is **singular**.
- **Identity matrix** $I = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$
  $AI = IA = A$
- Matrix A is **symmetric** if $A = A^T$

$$\begin{bmatrix} 1 & 2 & 9 & 5 \\ 2 & 7 & 4 & 8 \\ 9 & 4 & 3 & 6 \\ 5 & 8 & 6 & 4 \end{bmatrix}$$

Matrices

- **Inverse** of a square matrix $\boldsymbol{A}$ is matrix $\boldsymbol{A}^{-1}$ s.t. $\boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{I}$

- If $\boldsymbol{A}$ is singular or not square, inverse does not exist. Pseudo-inverse $\boldsymbol{A}^{\dagger}$ is defined whenever $\boldsymbol{A}^{T}\boldsymbol{A}$ is not singular (it is square)
  - $\boldsymbol{A}^{\dagger} = (\boldsymbol{A}^{T}\boldsymbol{A})^{-1}\boldsymbol{A}^{T}$
  - $\boldsymbol{A}^{\dagger}\boldsymbol{A} = (\boldsymbol{A}^{T}\boldsymbol{A})^{-1}\boldsymbol{A}^{T}\boldsymbol{A} = \boldsymbol{I}$

MATLAB

- Starting matlab
  - xterm -fn 12X24
  - matlab
- Basic Navigation
  - quit
  - more
  - help general
- Scalars, variables, basic arithmetic
  - Clear
  - + - * / ^
  - help arith
- Relational operators
  - ==,&,|,~,xor
  - help relop
- Lists, vectors, matrices
  - A=[2 3;4 5]
  - A'
- Matrix and vector operations
  - find(A>3), colon operator
  - * / ^ .* ./ .^
  - eye(n),norm(A),det(A),eig(A)
  - max,min,std
  - help matfun
- Elementary functions
  - help elfun
- Data types
  - double
  - Char
- Programming in Matlab
  - .m files
  - scripts
  - function y=square(x)
  - help lang
- Flow control
  - if i== 1else end, if else if end
  - for i=1:0.5:2 … end
  - while i == 1 … end
  - Return
  - help lang
- Graphics
  - help graphics
  - help graph3d
- File I/O
  - load,save
  - fopen, fclose, fprintf, fscanf